Intelligent Validation and Routing of Electronic Forms in a Distributed Work Flow Environment

MICHAEL COMPTON
SHAWN WOLFE
RECOM TECHNOLOGIES, INC.
ARTIFICIAL INTELLIGENCE RESEARCH BRANCH
NASA AMES RESEARCH CENTER
MAIL STOP 269-2
MOFFETT FIELD, CA 94035-1000

NASA Ames Research Center

Artificial Intelligence Research Branch

Technical Report FIA-93-31
October, 1993

Intelligent Validation and Routing of Electronic Forms in a Distributed Workflow Environment

Michael M. Compton
compton@ptolemy.arc.nasa.gov; (415) 604-6776

Shawn Wolfe
shawn@ptolemy.arc.nasa.gov; (415) 604-4760

Recom Technologies, Inc.
Artificial Intelligence Research Branch
NASA Ames Research Center, M/S 269-2
Moffett Field, CA 94035-1000

This paper describes a knowledge-based system for improving the efficiency of automated workflow systems by 1) ensuring the correctness and completeness of data contained on forms that are originated and transmitted electronically, and 2) generating an electronic "routing slip" that reflects who must approve the form. The system uses a form-independent validation engine and form-specific constraints to check that electronic forms are filled out correctly. If no errors are detected during validation, the system uses information on the form to generate a list of individuals and/or organizations that must approve it. This "approval path" information is added to the form so that it can be automatically routed from one approver to the next. The system is implemented in CLIPS and currently runs on Macintosh computers. It communicates with an off-the-shelf electronic forms package via AppleScript™ and can operate within the Apple Open Collaboration Environment (AOCE™), which supports a variety of other workflow capabilities including digital signatures, system-level electronic mail, and data encryption. The system has successfully validated and generated approval paths for approximately ten different types of forms, and is easily extended to new forms via a "BUILDCLASS" facility that automatically generates the CLIPS code necessary to represent and reason about the new form.

AI Topic:	knowledge-based systems
Domain area:	electronic forms, workflow automation
Language/Tool:	CLIPS, Informed, AppleScript, AOCE
Status:	Initial prototype complete, and being evaluated in the context of a site-wide workflow application
Effort:	Approximately 2 person-years of effort have been expended to-date
Impact:	Elimination of incorrect, incomplete, or incorrectly-routed forms is estimated to save tens of thousands of dollars per year in wasted effort.

To be presented at the Tenth IEEE Conference on Artificial Intelligence for Applications (CAIA-94) in San Antoino, TX in March 1994.

Background

The Ames Acquisition Division processes tens of thousands of purchase requests (PRs) each year. These PRs, which can be used for anything from procurement of computing equipment and laboratory instruments to allocation of funds for university grants, must be approved by anywhere from five to twenty people before they can be processed. More than half of these PRs need to be returned to the originator because they are incorrect or incomplete. Even when the PRs are filled out correctly, it's often unclear to the originator exactly who needs to approve a particular PR, and it is likely to be sent on to the wrong person. As a result, even a properly completed purchase request can take a very long time to get processed.

In 1991, the Artificial Intelligence Research Branch at NASA's Ames Research Center undertook an effort to determine whether AI technology could be applied to the validation and routing of an electronic purchase request. With assistance from the Information Systems Branch and the University Affairs Branch, development began on the Prototype Electronic Purchase Request (PEPR) system. The initial application of this system was to the PRs and associated evaluation forms that must be processed in order to fund external research through university grants. These types of PRs were chosen because of their relatively simple validation and routing requirements, and the fact that automation of this process would be of immediate value to the organizations involved. The system is now being extended to cover "small purchase" PRs, which require more complicated validation and more diverse approval paths.

Project Goals

In order to properly evaluate the suitability of knowledge-based technology to the validation and routing of electronic forms it was necessary to provide several other components of an automated workflow environment; namely, a high-fidelity rendering of the forms themselves, and an electronic mail system to transmit forms between users. It was also very important that these various software modules be able to communicate with each other as easily as possible, with little or no assistance from the user.

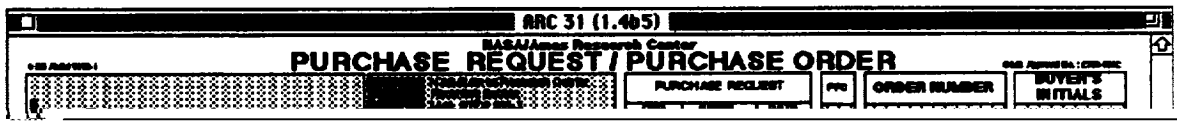
Our overall objectives were as follows:

- to develop a knowledge-based system that can
  - a) ensure that an electronic form is correct and complete,
  - b) generate an "approval path" that shows who needs to approve the form based on its content, and
  - c) interact "invisibly" with other components of an automated workflow system;
- focus on the Ames Purchase Request form (ARC31), but also be more generally applicable to other forms (i.e. minimize the system's dependence on a specific type of form); and
- provide enough of a complete end-user environment (electronic forms, electronic mail, etc.) to permit reasonable evaluation of the knowledge-based component.

Approach

The PEPR system is comprised of a commercial electronic forms package and a

design program, called Informed Designer, allows the forms designer to define the layout of the form, the appearance and behavior of data fields, and other aspects of the forms behavior. The "fill application" is called Informed Manager, and allows end-users to fill out the forms, print them, and/or mail them to other users. Informed not only meets all of the criteria outlined above, but also has built-in support for communication with a variety of commercial data bases and permits the embedding of scripts. Figure 2 shows an example electronic Ames PR form.



- that it be easy to integrate with the other tools (and customize, if necessary); and
- that it be readily available.

The "C Language Integrated Production System" (CLIPS) best fit our needs for the project. CLIPS was developed by the Software Technology Branch at NASA's Johnson Space Center (and was derived from the forward chaining component of a very successful commercial expert system shell). The fact that CLIPS comes with its source code enabled us to extend it to support Apple Events (the inter-program messaging facility that is built in to the Macintosh operating system with System 7). We also made CLIPS "scriptable" so that it could be controlled remotely and easily share data with the forms package.

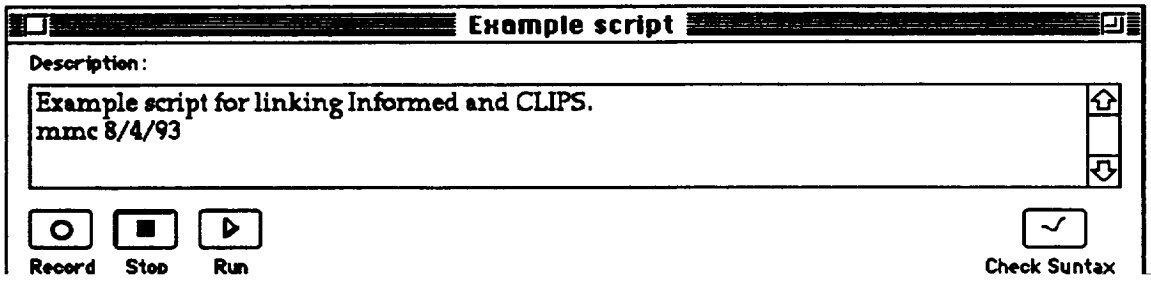
Integrating the Knowledge Base with the Forms Package

One of the most important goals of our system was that the CLIPS module be as "seamlessly" integrated with the other modules as possible. That is, we wanted the knowledge-based system's operation to be invisible to the user and have its output (the validation errors and the routing path) displayed to the user on the form itself.

The initial version of our system used a popular "keyboard macro" package to integrate the electronic forms and the knowledge base. By typing a single keystroke from within the forms application, the macro would cause the system to export the form's data to a disk file, switch to the knowledge system, read in the data from the exported file, analyze it, and print out the results of the analysis in the CLIPS "dialog window". While this approach worked, it had some serious drawbacks. First, because the keyboard macro simply simulated a user's input, the user had to watch a series of dialog boxes and simulated typing and menu commands as the macro executed. Second, the output of the knowledge based system was not easily returned to the forms application, and could only be printed out on the screen by CLIPS. Most importantly, however, was that integration by means of the keyboard macro package required that both applications be running on the user's computer. This meant that to actually put the system in front of real users would have required each user to be running their own copy of CLIPS and have the latest version of the knowledge base. This approach, of course, would have presented considerable maintenance headaches.

The current version of our system uses AppleScript, a newly-released scripting language for the Macintosh. It allows the two applications to communicate almost invisibly, and also allows the error messages or routing information to be inserted directly into the form itself. With the error messages contained in a field on the form, it becomes much easier for the user to correct the associated errors. When the list of required approvers for a particular form is contained in a field on the form, it is possible for an electronic mail system to automatically route the form to the proper people. Perhaps the greatest advantage, however, is that AppleScript can be used to control applications across an AppleTalk network. This enabled us to put a single copy of the CLIPS system on a centralized "server" machine, and allow

remote users to check forms that reside on their own computer, without requiring them to have their own copy of the knowledge-based system. This capability not only makes maintenance of the system much easier, but also allows the system to keep an accurate log of the validation or routing requests it receives. Figure 3 shows an example fragment of AppleScript code. (Note that this example AppleScript code is considerably simpler than that actually used in the PEPR system; it's included here merely as an illustration of how Informed and CLIPS can share data through the use of AppleScript).



functionality or support of new forms) by defining new specialized subclasses that inherit much of their functionality. The following is the (partial) class definition for the ARC31 form. Each slot corresponds to a field on the form. Fields that correspond to "table elements" on the form are represented as "multiple value" slots in the class definition. To facilitate the sharing of data between the forms package and the knowledge base, the slot names coincide exactly with the names of the fields on the form.

```
(defclass arc31 (is-a FORM)
      (concrete)
   (slot FORMTYPE)
   (slot ADPSIG)
   (slot ADPSIGDATE)
   (slot AMOUNT (multiple))
   (slot APPROPRIATION)
   (slot BRANCHSIG)
   (slot BRANCHSIGDATE)
   (slot BRIEFDESC)
   (slot BUYERSINIT)
   :
   :
)
```

Not all of the data entered onto the form is actually sent to CLIPS. Only the data from those fields that are referenced in the validation constraints (or routing rules) is required for the PEPR knowledge base to reason about the form.

Figure 4 shows the class structure of the PEPR knowledge base. The ORG class and its subclasses are used to represent the organizational structure at Ames. Instances of these classes (and their corresponding PERSON instances) are used in the construction of the approval paths. The ERROR and CONSTRAINT classes (and their subclasses) are used during form validation. Subclasses of the FORM object are instantiated and used throughout the knowledge base. The CLUE, PATH, and HYPOTHESIS classes are used to classify the data on the form prior to generation of the approval path (these objects are described in more detail in the sections that follow).

The greyed classes were added to support non-NASA forms used for other purposes in the pilot environment.

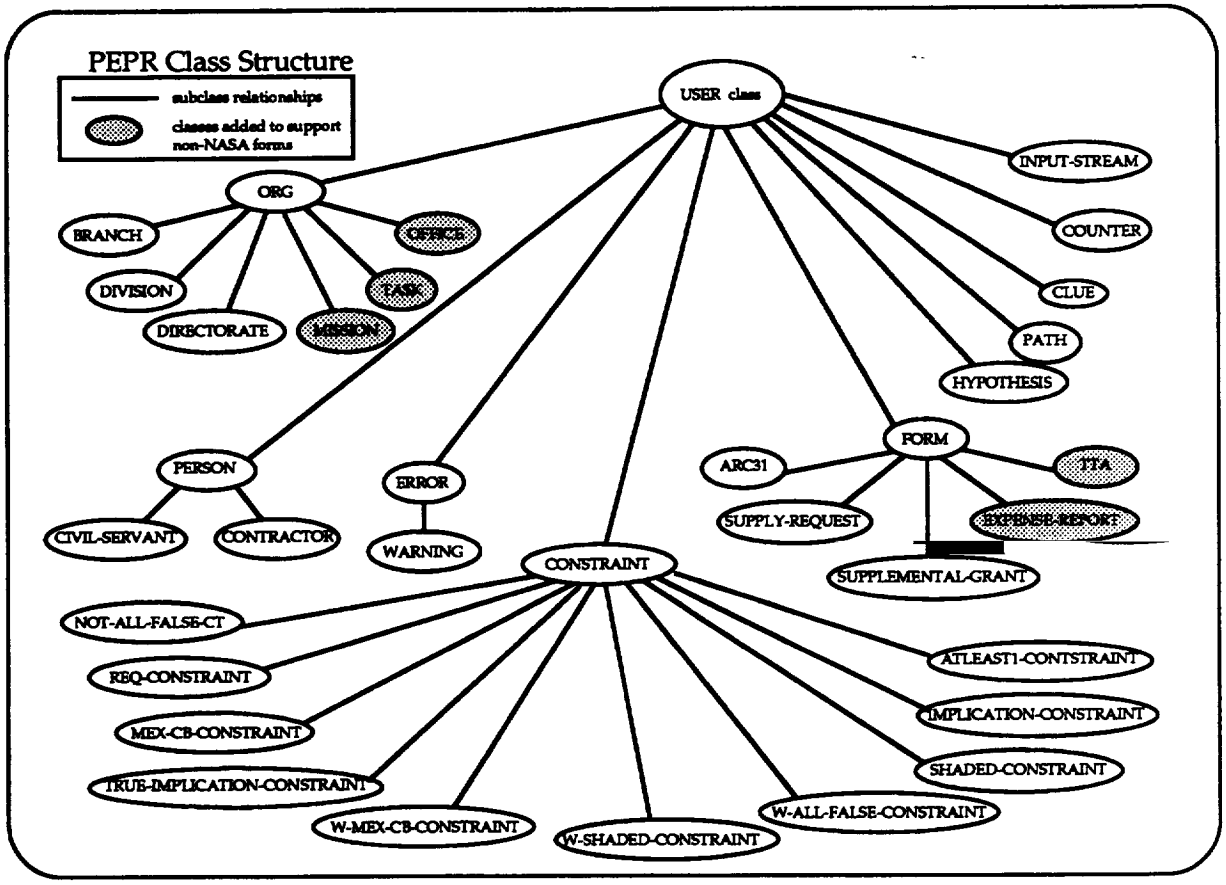


Figure 4 - The PEPR Class Structure

Validation Module

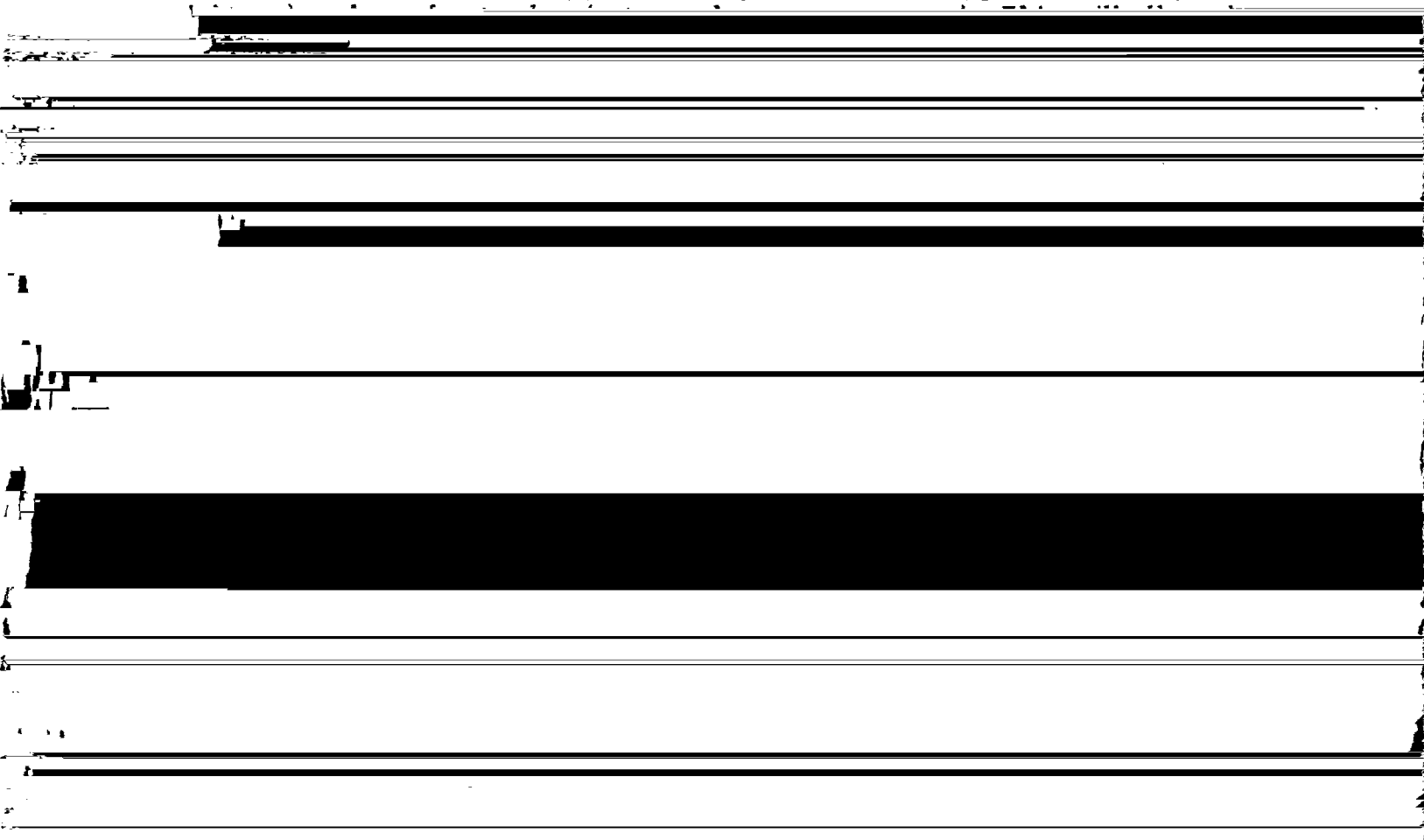
A completed form is considered "valid" if none of the constraints that are applicable to that form type are violated when applied to the form. When validation begins, the knowledge base creates an instance of the form being validated, collects values from all of the constrained fields on the form, and sets its instance's slots accordingly.

The following is a simple example of a constraint that reports an error when the user enters a value in a field that should be left blank:

```
([no-py-fields] of shaded-constraint
   (form arc31)
   (fields PY)
   (msg "No PY fields should be filled in."))
```

There are currently seven constraint subclasses, each of which has its own "apply" method that is executed during validation. In addition to these "error constraints", there are three types of "warning constraints" that can generate messages for the user but do not inhibit subsequent generation of an approval path. For instance, one page of the ARC31 (not shown) has a table of checkboxes with which the user can attempt to classify the items being ordered. When an apparent

inconsistency between the user's classification and the system's classification is identified, the user is presented with a warning message and can then decide whether to fix the inconsistency or proceed. Allowing these "soft" constraints allows the system to point out an apparent discrepancy without unduly limiting the user's intent. In the future, the system will support a "user-context" feature, which will allow constraints to be applicable only for a certain type of user (e.g., the



knowledge base to be applied differently for different types of users.

Classification Module

If the form is valid, the system invokes the classification module which determines whether the line items on the PR cause it to fall into any "special" categories. The system scans the description fields on the PR and compares them with the appropriate "clue" strings for each category. If a match is found, the system notes the "hypothesis" associated with that clue and records the level of belief associated with that clue. PEPR uses a certainty factor calculus derived from that of the MYCIN system, and can use the presence of a clue to indicate either a positive or negative belief that the corresponding hypothesis holds (i.e. that the PR falls into the associated category). This CF calculus dictates how evidence resulting from multiple clues is to be combined, and allows both for "positive" (supportive) clues and "negative" (refutive) clues with respect to a particular hypothesis. For example, the following clue indicates that the word "macintosh", when found in the BRIEFDESC field of the PR, indicates that the PR is for ADP equipment with a certainty factor of 90.

```
([arc31:ADP:BRIEFDESC:141 of CLUE
```

Classification of the form proceeds in a straightforward manner. PEPR iterates through the HYPOTHESIS instances, searching for each relevant keyword, and updating the measure of belief for the current HYPOTHESIS if the keyword is found. When this process is completed, a belief is asserted for each HYPOTHESIS instance that meets a minimum belief criterion.

Path Generation Module

Approvals that are required for an ARC31 form fall into two categories: management approvals and special approvals. Management approvals reflect the structure of the organization and the position of the originating organization therein. Special approvals are those that are required for particular types of items, regardless of who is ordering them.

At Ames, the ARC31 requires approval from each level in the management chain, from the originating organization up to through the Directorate above the originating organization. Depending upon the types of materials being ordered, the ARC31 may also need to be approved by a variety of special approvers. Finally, the form must be routed to the Purchase Control Unit, which is the ultimate recipient of all ARC31 forms generated at the Center.

Although ultimately represented as a single list of approvers, the approval path for an ARC31 is constructed from three distinct parts. The first part is a list of initial approvers. Each initial approver is either defined relative to the originating organization or is the same for all organizations. This list is sequential, for approval must follow a predetermined order. The next part is determined by the classification of the form. Depending upon how the form was classified (and in some cases the total dollar amount of the form), certain organizations will have to approve it. The process of adding these approvers is nondeterministic, and indeed the approvals could occur in parallel since none of these "special approvals" are dependent upon the others (however, we currently represent these approvers sequentially). The final approver is the recipient organization, which must be the last organization to approve the form (in the case of the ARC31, the Procurement Control Unit is always the final approver).

Generation of the approval path is determined by form-specific facts in the PEPR knowledge base. Since the initial approvers and the final approver are dependent upon the classification of the form, and since it is important to maintain their order, we use a single fact to represent them. This fact has a component that is a list of references which constitute the initial approvers, and another component containing a reference to the recipient organization. Each reference is either a pointer to a particular organization's object or a token that defines an organization with respect to the originating organization (e.g., DIRECTORATE, which would refer to the directorate that originating organization is in).

The special approvals that a PR requires need not occur in any particular order. Therefore, no ordering rules are applied to the special approvers (except that they must fall between the initial approvers and the recipient organization). Several facts can be used to represent the special approver information and the CLIPS pattern matching algorithm can add these approvers randomly to the list. Each special approver fact has three parts: a part that names the categorization to which

they are applicable, a reference to the special approver organization, and an optional

additional test (for example, a test to see if the dollar amount on the PR is over a certain limit).

Figure 5 shows an example routing slip for the Ames PR form (note that in this example, the form has already been sent to the first three approvers).

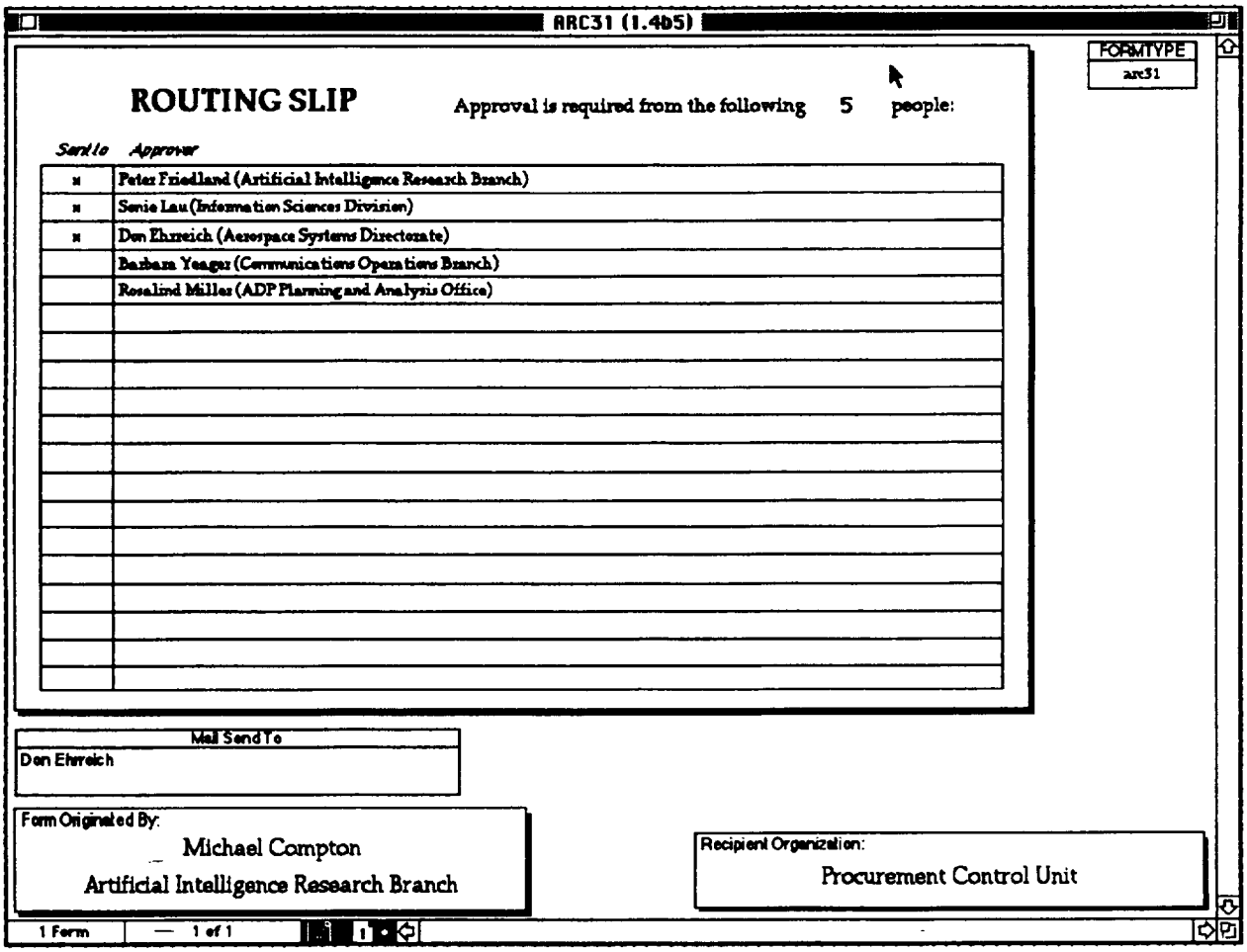


Figure 5 - An Electronic "Routing Slip" for the Ames PR Form

Applying PEPR to Other Forms

The PEPR system maintains form-specific data structures that enable it to validate and generate approval paths for several electronic forms. For the ARC31 form, these structures were hand-coded. To facilitate the application of the PEPR system to new forms, tools were developed that automate the procedures by which these CLIPS structures are generated. These procedures allow the new data structures to be created directly from the forms themselves. This has dramatically reduced the time required to adapt the PEPR system to new forms (from hours to seconds).

The 'BUILDCLASS' Facility

PEPR's representation of a form consists of a CLIPS class definition that includes a slot for every field on the form. The "BUILDCLASS" facility takes advantage of the fact that Informed Manager can export data from a form to a disk file in what is known as "merge format". In this format, the names of the fields are written to the file along with the data they contain. In order to have a new class definition generated automatically, the knowledge base developer need only export an empty form in merge format. This permits the BUILDCLASS module to read in the field names as if they were data; only instead of using this information to fill in the slots of a previously-defined form instance, it is used to build a "defclass" data structure that can then be saved in a knowledge base file and used later when a form of that type is to be validated.

The 'MAKECT' Facility

It can also be time-consuming to hand-code the CLIPS objects that comprise the validation constraints. Just as the BUILDCLASS facility can be used to automatically construct the class definitions, the MAKECT facility can be used to automatically construct constraint definitions.

The knowledge base designer uses the MAKECT facility by filling out the form to be checked with various codes in the fields to be constrained. For example, putting a single character 'R' in a text field will cause a "required" constraint to be generated. Likewise, putting an 'S' in a "shaded" field will generate a constraint indicating that a field should not be filled in by an end user. Similar codes have been defined for other fields on the form that can only accept numbers or dates. The knowledge base designer can also specify whether certain fields should be validated separately or collectively.

Once the proper codes are entered onto the form and exported to a disk file, the MAKECT facility reads in the file, parses it, and generates the proper CLIPS source code to represent the constraints.

Current Status

The PEPR system is currently being evaluated with respect to its usability in a production environment at NASA Ames. The knowledge base for the ARC31 form is being refined to conform to recent modifications in the Ames procurement processes, and later this year the system will be tested by real users within the Aerospace Systems Directorate.

With the upcoming release of Apple's Open Collaboration Environment (AOCE, due in the fall of 1993), the approval paths generated by the PEPR system will be used to directly control the routing of the forms through the approval process. Because AOCE also supports digital signatures and system-level access to electronic mail (among other services), the PEPR system will be functioning in a true distributed workflow environment.

Future Plans

There are three main areas in which we hope to be able to improve the PEPR system's functionality.

First, we want to determine whether machine learning techniques can be used to improve the classification module by automatically acquiring new knowledge about how to classify line items on a PR with respect to the special approvals they require. Currently these classification clues and hypotheses are hand-coded, and an automated approach to acquiring this knowledge could help improve the system's classification capability with minimal manual maintenance. We are also studying the issues involved with acquiring some types of classification heuristics directly from the end users of the system.

Second, we want to reduce the system's dependence on its internal data base of approvers and organizations. This type of data is much better suited to a more conventional data base that can be maintained externally to the knowledge base.

Third, we are studying ways by which the knowledge system can help the user correct errors that are identified during validation. This would involve augmenting the system's representation of error types with repair strategies that can automatically be pursued if the user indicates that he or she wants help with correcting the problems reported by the PEPR validation module.

We also are interested in applying the PEPR architecture to other types of

References

[1] Compton, M., Stewart, H., Tabibzadeh, S., Hastings, B. 1992 *Intelligent purchase request system,* NASA Ames Research Center Tech. Rep. FIA-92-07

[2] Shortliffe, E. H. 1976 *Computer-based medical consultations: MYCIN.* New York: American Elsevier.

[3] Hermens, L.A., Schlimmer, J.C. 1993 *Applying machine learning to electronic form filling.* Proceedings of the SPIE Applications of AI: Machine Vision and Robotics